1. Introduction
   1. Vision
   2. The Problem Domain
   3. Stakeholders
   4. Software Context
2. Functional Requirements
   1. Single-player game
   2. Network game
   3. Physics simulation
   4. Fleet management
   5. Variety of special environment conditions
   6. Change game settings
3. Non-functional Requirements
   1. Performance Constraints
   2. Reliability Constraints
   3. Portability Constraints
4. Testing and QA
5. Usage Scenarios
   1. User Profiles – The Actors
   2. Special usage considerations
   3. Use-cases
6. Plan for Software Iteration One & Risk assessment

Chapter 1

Introduction

* 1. Vision

The goal of this project is to create a working videogame, which can be worthy of commercial release with further development. The game will be a turn-based strategy game with its main sources of inspiration being the titles of Frozen Synapse, XCOM, FTL and Star Control.

What sets our game apart from the rest of the titles available today is the unique way in which we execute the turns. Usually turn-based games take the player’s order, perform it immediately, and then gives control to the opponent which can react by issuing his own orders (like chess). In our game each player gives his orders for the turn, and **both** of the players’ orders are executed simultaneously. This adds another layer of depth by asking each player to anticipate his opponent’s actions and plan accordingly.

* 1. The Problem Domain

-

* 1. Stakeholders

Customers: gamers who favor the strategy, turn-based tactics, Sci-Fi and roguelike genres.

Publishers: Development will be done independently, without a publisher while distribution will be done digitally, with possible retailers being Valve, Humble Bundle and GOG.

* 1. Software Context

When a user starts the software, he is greeted by a menu, from which he has a few options. The options include starting a new game, joining a game on a network, or exiting the program.

The software’s inputs are keyboard and mouse, interacting with a dynamic GUI. Whenever the user interacts with object in the GUI, the GUI is update to reflect the changes caused by the action.

Chapter 2

Functional Requirements

2.1. Single-player game

The user should be able to start a new game at any time, throwing any saved data. The opponent will be controlled by an AI player in this mode. The AI will be written by us from scratch.

2.2. Network game

The player should be able to play the game against a distant player over the network, without affecting his previous local games. We will take advantage of some of Unity’s standard libraries for network communication and synchronization.

2.3. Physics simulation

The game physics will be similar to the real-world physical laws. We will use some of Unity’s built-in physics engine in order to simulate some of the actions and help handling collisions.

2.4. Fleet management

The player will be able to issue commands to each unit in the fleet individually, and his orders will be performed simultaneously by both his fleet and his opponent’s fleet.

2.5. Variety of special environment conditions

Some of the battle arenas will have special properties which will be different from “normal space” and will change the way the player has to control his units significantly.

Examples:

* Zones containing solar-objects which will affect units through gravity
* Asteroid field which will have many moving asteroids will hurt units upon impact and block projectiles
* Dust Field which will change the way units and projectiles move within them

2.6. Change game settings

The user will be able to adjust several of the game’s settings, such as the way things are shown to the user, or how the game works. For example a user could change the window’s resolution, animation’s speed, music’s volume or UI displays.

Optional Requirements

* Self-learning AI player
* Full single-player campaign
* Tutorial
* Special scenarios

Chapter 3

Non-functional Requirements

* 1. Performance Constraints
* **Technical-** Basic unity requirements detailed here are required for running the game: <https://unity3d.com/unity/system-requirements>
* **Frame Rates-** The game should be running at 30-60 FPS on any PC meeting the technical requirements.
* **Response Times-** Delay for network game should be minimal and lower than 1 second when resolving turns. Single-player game should have no delays.
  1. Reliability Constraints
* The system should suggest the user to send a “bug report” when it encounters a failure.
* The system should encounter up to 1 fault per 10 hours of work on 99% of the computers.
  1. Portability Constraints
* Due to the fact that we are building the game in Unity we have very few platform constraints since it can compile the same code to the common 3 PC operating systems as well as to mobile OS's and consoles. However, we are being constrained by the physical input and visual devices of the different platforms. For optimal use we require the user to have access to a mouse, a keyboard and a decently sized screen, which is found on desktops and laptops.

Chapter 4

Testing and QA

Code testing will be written and run using the built-in unity testing suite, in a similar fashion to the way it is done in JUnit/NUnit.

For QA we will setup stations at conferences and at the campus, in which the game can be played by the public. During the sessions we will watch and take notes, and at the end of the play session we will ask for verbal feedback.

Chapter 5

Usage Scenarios

* 1. User Profiles – The Actors

Our actors in the system are the users.

* 1. Special usage considerations

--

5.3. Use cases:

User

Other User

Changes

system

settings

Start New Game

Two players start

a multiplayer game

Choose combat settings

Choose combat settings

Opening screen

User

Match

Left click

on object

Right click

Click on “Actions” button

Choose

action

Left click

Click on “End Turn” button

Press “Esc”

Changes

system

settings

Quit

Chapter 6

Plan for Software Iteration One & Risk assessment

TODO